A distributed architecture of Sensing Web for sharing open sensor nodes

著者	Kanbayashi Ryo, Sato Mitsuhisa
journal or publication title	Future generations computer systems
volume	27
number	5
page range	643-648
year	2011-05
権利	(C) 2010 Elsevier B.V. NOTICE: this is the author's version of a work that was accepted for publication in Future generations computer systems. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Future Generation Computer Systems, Vol.27 Issue 5, Pages:643-648. doi:10.1016/j.future.2010.11.022
URL	http://hdl.handle.net/2241/113804

doi: 10.1016/j.future.2010.11.022

A Distributed Architecture of Sensing Web for Sharing Open Sensor Nodes

Ryo Kanbayashi* and Mitsuhisa Sato

Graduate School of Science and Information Engineering, University of Tsukuba,
1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan,
{kanbayashi,msato}@hpcs.cs.tsukuba.ac.jp

Abstract. Sensing Web is a conceptual framework to shares sensors open in wide-area network with keeping privacy. Sensing Web targets large data such as image data or voice data, which may include privacy information. In this paper, we propose an architecture named SW-agent to realize the idea of Sensing Web. SW-agent protects privacy information with elimination of privacy information and appropriate access control. The elimination is done with data processing by remote execution program shipped to a node near a sensor. We found that SW-agent can execute remote execution program with up to 7% overhead in performance comparing its direct execution.

1 Introduction

Today, the need for real-world information is increasing rapidly. For example, services such as *Google Street View* [1] enable people to see selected street views on a web map without physically going to that street.

At the same time sensor devices such as video cameras, infrared sensors and microphones are being installed in buildings on roads and in station yards. Therefore, *Ubiquitous Sensor Networks*(USNs) are a promising technology for making use of these sensors. So far, however, only the owners of USN's have implemented major applications, and these closed networks are generally available only to the owners. For example, these are applications such as surveillance cameras in stores or cameras showing traffic flow with fixed point cameras, but the sensor data is available only to the respective owners and selected employees.

A concept called *Sensing Web*[2] is to enable people to share sensors openly in a wide-area network. The goal of the Sensing Web is to allow people to access actual sensor data in the same way they access the World Wide Web (WWW). In the Sensing Web, new applications or services using sensors can be created for implementation on both open and closed systems. Consider that you lose an object, a lost property finder service accessible with the Web browser through the Sensing Web might be used to find the missing object.

Different from existing sensor grids[3], Sensing Web is a open system. In addition, target data and target application are different. Most important requirement of Sensing Web is privacy protection[4]. Previous sensor grids target simple data such as the temperature and humidity data, which don't include

* Now at Fujitsu Limited. kambayashi.ryo@jp.fujitsu.com

privacy. In contrast, Sensing Web targets data such as image data or voice data, which includes many privacy information. Transmission of data including privacy information puts the information in danger. Therefore, an appropriate privacy protection method is needed. Handling data in network for acquisition of sensor data is also an important requirement. The data such as image data or voice data may be large, although the required information in the data may often be small. Transmission of all sensor data consumes network resources excessively. A method for acquisition of sensor data which consumes little network resources is needed. In addition, consideration of problems arise from several privileges of sensors and machines managing the sensors is needed. Since, unlike sensor grids, these resources of Sensing Web is offered by general public. Administration policies of these are also different from sensor grids.

We propose a distributed architecture named SW-agent as a prototype system of Sensing Web. Protection of privacy can be realized with elimination of privacy information and a access control mechanism. Elimination of privacy is done with data processing by remote execution program shipped to a node near a sensor. Reduction of communication traffic can also be done with elimination of useless data in a similar way.

Contribution of our research is following: First of all, we reveal issues of design for sharing sensor data on open systems, which will emerge in the future. We have designed and implemented a system called SW-agent. We develop a prototype of SW-agent and report its basic performance.

Remainder of the paper describes SW-agent and is organized as follows. Section 2 presents an overview of the Sensing Web. Section 3 presents the design of SW-agent, which is composed of program shipping facility, privacy protection model, and authorization for access control. Section 4 presents prototyping of SW-agent. Section 5 describes an overview of related work. Our conclusions and future work are discussed in Section 7.

2 Sensing Web Project

The Sensing Web Project[2] is a three-year project launched in the fall of 2007. Goal of the project is to develop information technologies necessary for sharing the data of USNs spreading across society openly like current WWW.

However, natures of information handled on the WWW and the Sensing Web differ. On the WWW, privacy information can be removed or otherwise protected because most of the information is entered by humans. In contrast, it can not be removed because the information is actual sensor data automatically collected by sensors. Existing sensor usage, including sensor grids, need not to consider privacy protection because these are closed system. However, privacy protection should be considered as mandatory on Sensing Web because it is open system. Focusing on handling of privacy information is a notable characteristic of Sensing Web.

Realization of following elemental technologies is important on the Sensing Web.

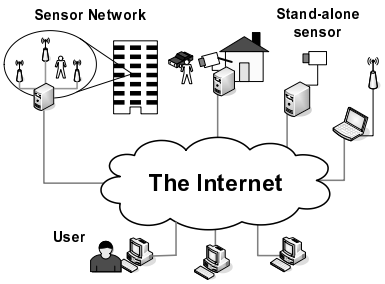

Fig. 1. A typical environment

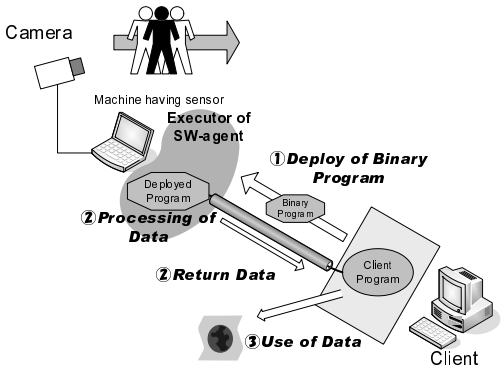

Fig. 2. Program Shipping Facility

- Sharing Sensor Data through Service/Request Matching: The information user requested is offered by matching the request and sensor data.
- Privacy Information Management in Sensor Data Acquisition: Privacy information in sensor data is handled properly on acquisition of sensor data. When sensor data is acquired, it is handled in symbolized form including no privacy information, which protects privacy information from leaking the information to malicious people.
- Information Integration for Presenting the Real World Information: Actual sensor data acquired from real world is integrated and offered to users as appropriate applications.

Our research develops a system, which realizes Sensing Web. Especially, solution to privacy information management in sensor data acquisition is offered.

3 Design of SW-agent

In this section, we present the design of SW-agent as a distributed architecture to realize the concept of Sensing Web. SW-agent is composed of two elements: program shipping facility, privacy protection model.

3.1 Architecture for Sensing Web

Figure 1 shows a typical environment of the Sensing Web, which currently is composed of closed sensor networks and standalone sensors such as Web cameras. Because these sensors are managed by several PCs connected to the Internet, these sensors are accessible through the Internet.

From the design's point of view, the Sensing Web is characterized by two property: privacy information and large sensing data. Although several sensor grids[3] were already proposed, it targets simple data such as the temperature and humidity data, which don't include privacy. The Sensing Web targets image data and human voice data, which may include many privacy information[4]. The size of data acquired by intelligent sensors such as video cameras may be large and often the data includes many useless information.

To realize the concept of Sensing Web, we propose SW-agents as a distributed architecture for Sensing Web. The SW-agent is an agent program installed near to the sensor to execute applications handling the sensor data. The system administrator sets up the SW-agent near to the sensor. At a minimum, directory service providing information about a sensor's location, user authentication and user authorization are needed for access to sensors.

The application developer uploads his or hers application handling the sensor data to the SW-agent by *Program Shipping Facility*. The client application communicates with its shipped application program to provide a service.

The SW-agent provides several methods to protect privacy information by monitoring the access to the device and the traffic to the client through the network.

Prior to the use of sensor data, sensor data were typically accumulated in storage and then accessed in sensor grids. However, as stated above, the required information in the sensor data is relatively small in comparison with the total amount of sensor data. Transmission of whole data consumes network resources excessively because the amount of whole data, such as images from video cameras, is large. Furthermore, the people captured in images may not prefer accumulation of the sensor data out of concern for the leaking of privacy information. Therefore, the Sensing Web handles requested data by data processing near sensors without accumulating and storing data to storages.

3.2 Program Shipping Facility

The *Program Shipping Facility* allows a binary program of an application to be deployed near a sensor, shown in Figure 2. Arrows on Figure 2 represents flow of processed sensor data. We describe the deployment as "program shipping". Program shipping has following advantages:

- Flexible data processing: Users can process sensor data flexibly through familiar program languages. The flexible data processing extract required information only out of a large amount of sensor data.
- Utilization of existing code: A program using a sensor can be used with little change. This advantage is important because most of users will have their own application using a sensor in a diffusion process of the Sensing Web.
- Usability: Users can use languages familiar to them. That is, users do not need to learn a special language.

A binary program is deployed through a procedure of web services, which make flexible deployment such as automatic deployment based on the algorithms possible.

Sensor information from the shipped program can be acquired by two means. The first is acquisition of data by web services. Users can acquire the data through pre-defined web service procedures. This technique is useful for acquisition of small amounts of data, such as the coordinate data from the lost property finder service. A flexible and user-customizable data access interface can be realized on web services because an arbitrary procedure can be defined. Flexible data access interface bridges the gap between sensor data and request of user. In

addition, the use of a web service enhances the compatibility of the Sensing Web with the World Wide Web. In addition, use of web service enhance compatibility of Sensing Web between Web. The second way is to use stream communication. If the size of processed data is large or data transfer is performed continuously, stream communication should be used because the overhead of remote procedure calls becomes a problem.

The SW-agent also performs sandboxing for the shipping program. Although shipping is performed by authorized users only, the machines managing the sensors used by the shipping program need ready protection from possible malicious users. Otherwise, the sensors and machines for shipping are not contributed by the owner. Therefore, sandboxing is important for the Sensing Web.

SW-agent also provides virtual devices abstracting the sensor devices. Consequently, users can acquire sensor data with a unified procedure using a virtual device without concern for the differences between environments.

3.3 Privacy Protection Model

The SW-agent protects privacy by running the remote execution program near sensors. Users can access all sensor data by deployment of a program, but users also are obligated to eliminate privacy information before sending data to a client through a network. The elimination is achieved through symbolization of sensor data. For example, we show protection on an application that surveys people passing on a road. The protection is achieved by following.

1. Authorization of User: A user who wants to deploy a program must acquire authorization from an access control service of the SW-agent.
2. Deployment of Program: The user deploys a binary program to a machine near a sensor, which manages the sensor. Then, the deployed binary program is started by the executor of SW-agent.
3. Elimination of Privacy Information: The program applies image processing to the image data acquired from the surveillance camera and outputs the number of passages per unit time.
4. Access to Privacy Eliminated Data: The user uses the processed information, which does not include privacy information.

In this application, video data from the surveillance camera may include confidential information, such as images of pedestrians' faces. However, this information is eliminated with symbolization by the shipping program. Therefore, the proposed model can enable a user to get sensor data while protecting the privacy of the sensor data.

However, if an inconsiderate user is authorized, data including privacy information may be sent to to the general public because the proposed model allows an authorized user to acquire all sensor data. Therefore, proper authorization is important in the SW-agent.

3.4 Authorization for Access Control

First of all, it is important who have the authority of access control for privacy information. Naturally, captured person, who privacy information belongs

to, should be able to authorize. However, authorization by captured people is difficult, which needs the people to carry a special device which notices sensor authorized users or interaction between the people and sensors. Therefore, in SW-agent, owners of sensors authorize the accesses instead of captured people. For the representation, SW-agent targets sensors whose owner can represents captured people such as cameras on home.

Appropriate authorization model is also needed. Authorization model for Sensing Web should satisfy followings;

- Performance: Processing costs needed for the model should not be large.
- Management Costs: Management costs for the model should not be high.
- Usability for Users and Owners: Both users and sensor owners should be able to apply for authorization without encountering problems.
- Transparent Use of Sensors Scattered across Different Organizations: Sensors used in the Sensing Web are scattered across different organizations. Users should be able to use these sensors transparently.

In the Sensing Web, sensors are classified into the following two units.

- Sensor Network: Sensors belonging together, which are managed as a sensor network. Typically, the sensors are owned by an organization such as a research institute or a company.
- Standalone Sensor: A sensor exists singularly. Typically, these sensors are owned by an individual.

Virtual Organization (VO) [5] is a promising model for the management of resources by grouping the resources. Grouping sensor networks by VO enables users to use sensors scattered across different organizations transparently. However, VO can't be applied to standalone sensors because there is no administrator for standalone sensors. Each standalone sensor is managed by an individual. Therefore, separate authorization is needed for each standalone sensor.

Authorization for standalone sensors may lead to much extra work for users. For example, if a user needs to acquire authorization from each site in order to run an application, which uses hundreds of sensors, the user has to send e-mails to hundreds of administrators or access hundreds of web portals to create an account. Therefore, an authorization model with few or no manual steps is needed for standalone sensors.

For realization of separate authorizations which need only a few manual steps, the *Policy Based Model* and *Chain of Trust Model* can be used. the Policy Based Model authorizes users by matching the policies of sensors with the attributes of users. The authorization is performed without any extra procedures by the user. However, the expressiveness of the policy is limited by the expressiveness of the attributes the users list, but a listing of attributes which allow arbitrary policy decisions is difficult. The Policy Based Model is suitable for authorization which needs a broad level of permission control, but it is not suitable for a precise level of control.

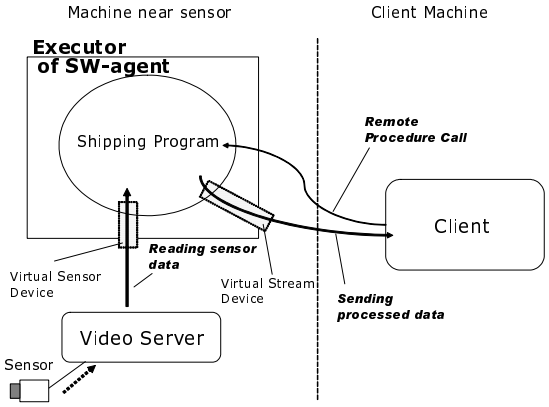

Fig. 3. Overview of executor part of SW-agent

The Chain of Trust Model authorizes users by a chain of trusts based on the following rule: "friends of my friend are trustworthy." A representative model of the Chain of Trust is *Pretty Good Privacy* (PGP) [6], which can authenticate users by a chain of trusts. For example, a user who has been allowed to access sensor-A on a building can access sensor-B on the same building. However, the effectiveness of the Chain of Trust Model depends on the activity level of owners of the sensor.

The models described above can be used simultaneously in a mutually complementary manner. In the future, we are planning the simultaneous use of multiple models based on the needs of the community.

4 Implementation of SW-agent

We implemented a prototype system of the SW-agent. Primarily, the execution functionality for program shipping is implemented. Figure 3 shows an overview of the execution environment.

The prototype is implemented based on BEE[7]. BEE emulates system calls in different operating systems (OSs) such as Linux and Windows, which enables a user to run a Linux binary on a different OS. Together, BEE and the mechanism for deployment of an execution binary enable the operation of program shipping. However, the prototype does not have a deployment mechanism and can only run on Linux at this time. Using dynamic class loading and the sandboxing mechanism of Java may be used for realization of remote execution. However, Java programs consumes more memory, thus limiting the scalability of the Sensing Web, which runs many processes on a machine. In addition, useful native code libraries such as *Open Computer Vision Library* (OpenCV) [8] cannot be used with implementations constructed with Java.

In the following sections, we describe our implementation of the SW-agent in more detail.

4.1 Sandboxing for Secure Execution

Sandboxing is achieved by hooking system calls using the *ptrace* system call in Linux. SW-agent changes the system call handler by *ptrace*, and forces system calls to be handled by the modified call handler by modification of the stack data. The modified handler determine whether each call is permitted with predefined policies. If a call is permitted, the handler calls the original system call using passed arguments. Otherwise, the handler blocks the request and returns an error code. Currently, SW-agent supports *open* system call.

4.2 Virtual Devices

The prototype has two virtual devices: a *Virtual Sensor Device* and a *Virtual Stream Device*. Virtual devices are realized by system call emulation, which changes the system call handler with *ptrace*, as in sandboxing. We implemented only the essential system calls: *open*, *read*, *write*, *close*, *ioctl*.

Virtual Sensor Device enables the user to access a sensor device transparently. The prototype supports emulation of a video device. We implemented the essentials of the video4linux [9] API interface, which is a standard interface for accessing video devices. For supporting video4linux interfaces, we implemented emulations of *ioctl*. Virtual Sensor Device is exported as "*/dev/sensor*." A program can read data by calling the *read* function for "*/dev/sensor*." Virtual Sensor Device acquires sensor data from a video server. The video server provides sensor data read from a real video device to Virtual Sensor Device through interprocess communication. The video server is implemented as a process separate from the executor. While normal device access uses a video device exclusively, our implementation enable users to share a sensor device simultaneously.

Virtual Stream Device realizes easy and efficient data sending to multiple clients. Virtual Stream Device abstracts the stream channel to clients and is exported as "*/dev/client*." Currently, the prototype supports sending data to a clients only.

Unlike the specification of web services for stream data such as MTOM/XOP [10][11], a user can set socket parameters for the Virtual Stream Device. This capability means communication can be adopted to a network environment. Furthermore, flexible transfer such as third-party transfer is possible by switching transfer destinations.

4.3 Remote Procedure Call Interface

We used *Simple Object Access Protocol* (SOAP) for realization of remote procedure call. SOAP supports not only primitive numeric values but also arrays and structured data as the arguments and return values, which enables flexible data access. Most codes for realizing remote procedure calls can be generated by our generation tools. Users only have to write the definitions and implementations of the procedures. We used the gSoap [12] for implementation. The code generation tools invokes tools of gSoap internally.

4.4 User Scenario

In this section, we present the user scenario of SW-agent. An execution binary for shipping is created by the following steps.

1. Describe Procedure Definition: A user describes the definition of a remote procedure call of web services with the description format of gSoap. In gSoap, a definition is described as a prototype definition on a header file.
2. Generation of Skelton Code: The user generates the skeleton code of a remote procedure call with the generation tool of gSoap.
3. Implementation of Procedure: The user writes the implementation of the generated skeleton code.
4. Generation of Execution Binary: The user makes an execution binary by linking the object file generated from the implemented code and our offering object file. Our offering object file includes the main function and something. Linking should be static due to the constraints of BEE.

Deployment of the execution binary is also performed through a remote procedure call of the web service. Currently, the prototype does not support deployment of an execution binary. We are planning to offer a client-side tool that deploys an execution binary through SOAP.

5 Performance Evaluation and Experiment

In this section, we report the results of an experiment and a performance evaluation of the SW-agent. The objective of this evaluation and experiment is to examine availability and especially performance characteristic of shipping program facility. First, we examined the basic performance of shipping program facility. Second, we conducted a sandboxing experiment.

• **Basic Performance of System Call Emulation:** SW-agent executes most of a codes directly on processors, except for system calls. Therefore, a program which has a system call is executed with the overhead of the system call hooking.

First, we examined the overhead of system call hooking. The overhead is the average execution time acquired by measuring the time of the *getuid* system call. We calculated the average execution time by executing the system call 10,000 times on and not on SW-agent. A machine used for evaluation had Core2Duo 2.4GHz and 2GB memory, and ran Linux Kernel 2.6.24.

As a result, the direct execution time was approximately 0.25 μsec. The execution time of SW-agent was approximately 15 μsec. This result means that overhead of SW-agent is approximately 15 μsec. The impact of this overhead on applications is examined in the following evaluations.

• **Performance of Virtual Sensor Device:** We have examined performance of Virtual Sensor Device. The performance is the average time calculated by measuring the time required to acquire 10,000 frames. We measured the execution time by using both the real sensor device directly and the Virtual Sensor Device. The frame size was 640x480. A machine used for evaluation was same as prior evaluation. Capturing board attached to the machine is Buffalo CBP-AV, whose frame-rate is 30fps and resolution is 640x480.

As a result of the experiment, the execution time required to acquire frame data was approximately 33375 μsec by direct access and approximately 33376 μsec by the Virtual Sensor Device. This result means that the overhead of the

Table 1. Evaluation Configurations

Name	Machine	Network
Local	Machine-B to Machine-B	omni.hpcc.jp
LAN	Machine-A to Machine-B	omni.hpcc.jp
WAN	Machine-A to Machine-B	hpcs.cs.tsukuba.ac.jp to omni.hpcc.jp

Table 2. Execution Time of Remote Procedure Calls

Network	Procedure	Native	SW-agent	Overhead
Local	Procedure-A	31 μsec	1117 μsec	351%
	Procedure-B	35 msec	37 msec	107%
LAN	Procedure-A	1103 μsec	1449 μsec	131%
	Procedure-B	36 msec	38 msec	106%
WAN	Procedure-A	6903 μsec	7396 μsec	107%
	Procedure-B	78 msec	79 msec	102%

Virtual Sensor Device is approximately 1 μsec. This overhead is because of system call hooking by *ptrace* and inter-process communication between the video server and the executor part of the SW-agent. This overhead is smaller than that of a former evaluation. We assume that this decrease occurs because the Virtual Sensor Device omits some device calls which the direct access needs, such as the order of starting capture. The overhead of the Virtual Sensor Device is sufficiently small.

• **Performance of Remote Procedure Call:** We examined the performance decline of the remote procedure call resulting from system call emulation. We measured the execution time on and not on SW-agent, as well as the execution time on the three transition paths shown in Table 1. In the evaluation, we used two procedures: procedure-A, procedure-B. Procedure-A has no arguments and returns an integer value. Procedure-B has no arguments and returns 900 Kbytes of data, equivalent to 640x480 video frame data. The communication data between the shipping program and the client is base64 encoded and then transmitted as XML messages using the HTTP protocol. For examination of average execution times, we executed procedure-A 10,000 times and procedure-B 100,000 times. The network latency on the WAN was approximately 2 msec. For the examination, we used two machines: Machine-A, Machine-B. Machine-A had Core2Duo 2.2GHz, 2GB memory, and Gigabit Ethernet, and ran Linux Kernel 2.6.24. Machine-B had Xeon 3.0GHz, 2GB memory, and Gigabit Ethernet, and ran Linux Kernel 2.6.9. Version of gSoap library used for implementation of web service is 2.7.

Table 2 shows the execution time of each configuration and the ratios of the overhead. The result shows that the overhead of procedure-A is greater than that of procedure-B. This is because the system call dominates a larger part of the total execution time in procedure-A. The result also shows that a larger network environment has smaller overhead with both procedures. This is because the overhead of system call hooking becomes smaller as the data transition time

increases. The WAN performance is important in the Sensing Web, which shares sensors on the Internet. The overhead on WAN is sufficiently small.

• **Basic Performance of Stream Data Transmission:** We evaluated the stream data transmission between the shipping program and client. The performance of stream data transmission is dependent on the data transmission bandwidth. We measured the bandwidth of the Virtual Stream Device and MTOM/XOP on LAN and WAN.

On burst transfer through a raw socket, the max bandwidth for LAN was approximately 871 Mbps and approximately 134 Mbps for WAN. As a result of the measure, the max bandwidth of the Virtual Stream Device for LAN was approximately 876 Mbps and approximately 134 Mbps for WAN. The max bandwidth of MTOM/XOP for LAN was approximately 860 Mbps and approximately 128 Mbps for WAN. This result shows that the performance of both stream transmissions was almost equivalent to a normal raw socket.

6 Related Work

There are many works which deal sensor networks[13–18]. However, most of these works target special sensor device such as mote[19], which can work autonomously with a simple processor, small amount of memory, and a wireless network. Therefore these researchs differs in hardware constraints and data characteristic from our project which targets sensor devices such as video camera and microphones, which produces large amount of media data.

Hourglass[20] aims to integrate multiple sensor networks. The integration enable people to use sensors on multiple sensor networks, which are scatterd across different organizations and geographically separated, trnsparently as well as SW-Agent. Hourglass deploys its compornents which collaborate with other compornents on sensor networks. The deployed compornents make up the data flow path of sensor data. When size of sernsor data is large, they can be reduced by deployment of some compornents which process the data on data flow path. Cougar[21] provides DB like access interface to user, which hides difference between multiple sensor networks. Sensor data is acquired by querying using special query language, which is distributed over the sensor networks if needed. However, these does not consider privacy problem and authentication. Unlike our research, Cougar targets special devices such as mote and does not consider conpatibility with World Wide Web.

There has already been a proposal to utilize the data acquired from sensors installed in the real world. Open Geospatial Consortium (OGC) proposed *Sensor Web Enablement* (SWE) [22], which is embraced in several projects. Buyya et al. proposed *Open Sensor Web Architecture* (OSWA) in [3] and implemented it in [23]. IrisNet[24] aims to realize *worldwide sensor web* which integrates commodity off-the-shelf sensor devices such as Web cameras. However, these can not process stream data efficiently because they acquire sensor data by query language with poor expressiveness that for processing stream data. Therefore, they cannot avoid sending all the data to a client. In addition, OSWA and IrisNet does not consider privacy. In contrast, SW-agent can process stream data

efficiently with remote execution program and has a mechanism for protecting privacy.

Some systems can already realize remote program deployment [25]. However, most of these can run a system-specific binary only. Users are forced to learn a system-spennnnncific rule to prepare the program. Therefore, these cannot be used for realization of the Sensing Web, which is targeted toward casual sensor sharing.

Issues of privacy information in pervasive computing are discussed on [4, 26–28].

7 Conclusion and Future Work

In this paper, we described an overview of Sensing Web and presented that the research issues for realizing it are privacy protection and consumption of communication resources. We proposed a architecture named SW-agent. SW-agent can resolve the issues by shipping a program, which eliminates privacy information and needless data, into the node near a sensor and with access control based on the authentication mechanism.

We implemented a prototype system, tested the sandboxing function and then evaluated the basic performance. The results of our examination showed that SW-agent can execute remote execution program with up to 7% overhead in performance comparing direct execution, which is acceptable for Sensing Web.

In the future, we intend to work on the following:

- Currently our prototype has no authorization system and deployment system. We plan to construct an appropriate authorization model based on the needs of a community and implement an authorization system based on it. In addition, we plan to implement a deployment service. After these, demonstration experiments for validation of architecture and each models of SW-agent. In the experiments, validation in the perspective of user will be commited.
- Mutual use of processed information between shipping programs may achieve more flexible and efficient sensor use in terms of both usability and network resource consumption. We plan to investigate it with a work-flow model on mutually connected shipping programs.

Acknowledgment

We would like to thank Dr.Yuich Ota and Dr.Itaru Kitahara, and Takashi Tsushima (Graduate School of Systems and Information Engineering in University of Tsukuba) for technical advices and supports. The authors achnowledges the contribution of all the members of the Sensing Web Project. The present study was supported by Effective and Efficient Promotion of the Coodination Program of Science and Technology Projects in the Special Coodination Funds for Promoting Science and Technology, which is conducted by MEXT of Japan, and Japan Science and Technology Agency(JST).

References

1. Google Street View: http://www.google.com/help/maps/streetview/.
2. M. M. et al, Sensing Web Project - How to handle privacy information in sensor data - (June 2008).
3. C. khong Tham, R. Buyya, SensorGrid: Integrating Sensor Networks and Grid Computing, in: Special Issue on Grid Computing, 2005.
4. P. Bhaskar, S. I. Ahamed, Privacy in Pervasive Computing and Open Issues (2007).
5. I. F. et al, Physiology of the Grid:Making the Global Infrastructure a Reality, Wiley, 2003, pp. 863–869.
6. Open PGP: http://www.openpgp.org/.
7. Y. Uemura, Y. Nakajima, M. Sato, Direct Execution of Linux Binary on Windows for Grid RPC workers (March 2007).
8. Open Computer Vision Library: http://sourceforge.net/projects/opencvlibrary/.
9. video4linux: http://linux.bytesex.org/v4l2/.
10. SOAP Message Transmission Optimization: http://www.w3.org/TR/soap12-mtom/.
11. XML-binary Optimized Packaging: http://www.w3.org/TR/xop10/.
12. The gSOAP Toolkit for SOAP Web Services and XML-Based Applications: http://www.cs.fsu.edu/ engelen/soap.html.
13. Y.Yao, J.E.Gehrke, Query prcessing in sensor networks, In First Biennial Conference on Innovative Data Systems Research(CIDR 2003).
14. S. Madden, M. J. Franklin, J. M. Hellerstein, W. Hong, The design of an acquisitional query processor for sensor networks, in: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, SIGMOD '03, ACM, 2003, pp. 491–502.
15. P. Levis, N. Patel, D. Culler, S. Shenker, Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks, in: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1, USENIX Association, 2004, pp. 2–2.
16. J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, D. Ganesan, Building efficient wireless sensor networks with low-level naming, SIGOPS Oper. Syst. Rev. 35 (2001) 146–159.
17. W. R. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, Hawaii International Conference on System Sciences 8 (2000) 8020.
18. D. B. Johnson, D. A. Maltz, J. Broch, Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks.
19. J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, System architecture directions for networked sensors, SIGOPS Oper. Syst. Rev. 34 (2000) 93–104.
20. J. Shneidman, P. Pietzuch, J. Ledlie, M. Roussopoulos, M. Seltzer, M. Welsh, Hourglass: An infrastructure for connecting sensor networks and applications, Tech. rep. (2004).
21. Y. Yao, J. Gehrke, The cougar approach to in-network query processing in sensor networks, SIGMOD Record 31 (2002) 2002.
22. G. Percivall, C. Reed, OGC Sensor Web Enablement Standard, Vol. 9 of Sensors & Transducers, 2006, pp. 698–706.
23. X. Chu, Open Sensor Web Architecture:Core Service (December 2005).
24. P. B. Gibbons, B. Karp, Y. Ke, S. Nath, S. Seshan, Irisnet: An architecture for a worldwide sensor web, IEEE Pervasive Computing 2 (2003) 22–33.

25. S. Brown, C. J. Sreenan, Updating software in wireless sensor networks: A survey, Tech rep ucc-cs-2006-13-07, Dept. of Computer Science, University College Cork, Ireland (2006).
26. M. Langheinrich, A privacy awareness system for ubiquitous computing environments.
27. M. Gruteser, G. Schelle, A. Jain, R. Han, D. Grunwald, Privacy-aware location sensor networks, in: Proceedings of the 9th conference on Hot Topics in Operating Systems - Volume 9, USENIX Association, 2003, pp. 28–28.
28. H. Chan, A. Perrig, Security and privacy in sensor networks, Computer 36 (2003) 103–105.